Expert Systems with Applications 38 (2011) 9325–9333

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Research of fast SOM clustering for text information

Yuan-chao Liu *, Chong Wu, Ming Liu

Department of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China

ARTICLE INFO

Keywords:
Self organizing maps
Text mining
Clustering efficiency
Feature coding
Similarity computation

ABSTRACT

The state-of-the-art text clustering methods suffer from the huge size of documents with high-dimensional features. In this paper, we studied fast SOM clustering technology for Text Information. Our focus is on how to enhance the efficiency of text clustering system whereas high clustering qualities are also kept. To achieve this goal, we separate the system into two stages: offline and online. In order to make text clustering system more efficient, feature extraction and semantic quantization are done offline. Although neurons are represented as numerical vectors in high-dimension space, documents are represented as collections of some important keywords, which is different from many related works, thus the requirement for both time and space in the offline stage can be alleviated. Based on this scenario, fast clustering techniques for online stage are proposed including how to project documents onto output layers in SOM, fast similarity computation method and the scheme of Incremental clustering technology for real-time processing, We tested the system using different datasets, the practical performance demonstrate that our approach has been shown to be much superior in clustering efficiency whereas the clustering quality are comparable to traditional methods.

1. Introduction

In an effort to keep up with the tremendous and ever-growing amounts of electronic documents from World Wide Web and digital libraries, people are much interested in developing technologies to organize and navigate massive text data for searching and exploring data, thus make it easier for end users to find the information they want efficiently and accurately. Data clustering is one important technique for grouping similar data items together for convenient understanding.

By organizing massive texts into some meaningful clusters, people can observe these data from a high-level point of view. In many occasions, people's requirements cannot be formulated clearly. Whereas by examining well organized structure of massive data from a high level, they may find something valuable or something unknown before. Thus it is convenient for discovering new interest and narrows searching range, and alleviates reading labor. When applied to textual data, clustering methods try to identify inherent groupings of text documents so that a set of clusters is produced in which clusters exhibit high intra-cluster similarity and low inter-cluster similarity (Cios, Pedrycs, & Swiniarski, 1998; Lois, Olivier, & Francois, 2007).This problem has received a special and increased attention from researchers in the past decades (Aliguliyev, 2009; Saraçoğlu, Tütüncü, & Allahverdi, 2007; Saraçoğlu, Tütüncü, & Allahverdi, 2008).

The focus of this paper is on fast clustering technologies of documents. Conventional data clustering methods, including agglomerative hierarchical clustering and partition based clustering algorithms; frequently perform unsatisfactorily for large text collections, since the computation time of many text clustering methods increase very quickly with the number of data items. Poor clustering results degrade intelligent applications such as event tracking and information retrieval. Most text clustering systems are less efficient because of two reasons: (1). the time cost of similarity computation. Similarity computation is the crucial component for text clustering system, and is usually done in high-dimension space, as documents are usually coded into high-dimension vectors by VSM (Vector Space Model) (Aliguliyev, 2009; Lee & Yang, 2009; Salton, wong, & Yang, 1975; Song, Li, & Park, 2009); (2) the frequency of similarity computation. Different from text categorization technology, similarity computation for text clustering system is more frequent, when the size of documents increases, the time complexity and storage space consumed become more and more intolerable for users. In fact, for many applications such as news group filtering, text crawling, and document organization, fast clustering and organization of text documents are required to process data in a timely fashion (Aggarwal et al., 2006; Liu, Cai, Yin, & Huang, 2006).

In this paper, we use Kohonen's Self Organizing Map (Kohonen, 1982), which has been widely used for text clustering (Chow, Zhang, & Rahman, 2009; Hung, Chi, & Chen, 2009), as the main framework. As similarity computation is very crucial for text clustering, and has much impact on clustering efficiency, we propose

* Corresponding author.
E-mail address: ycliu@hit.edu.cn (Y.-c. Liu).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.01.126

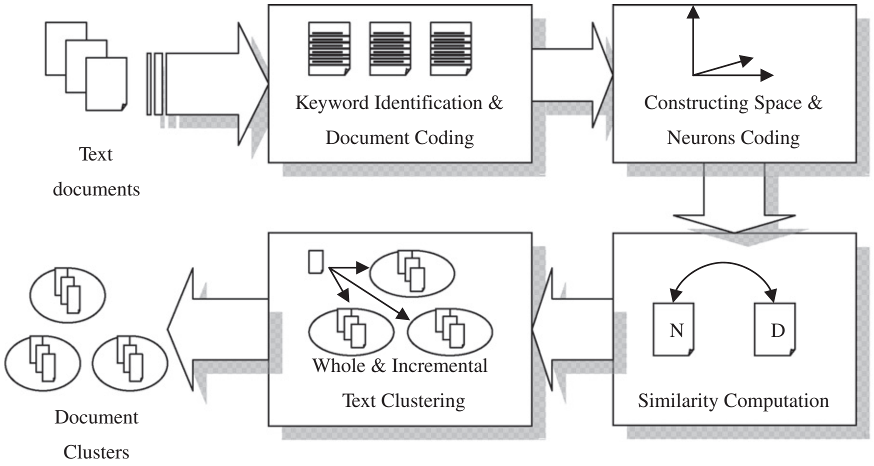

Fig. 1. The main framework of our system.

one novel feature representation and similarity computation method to make SOM text clustering much faster. Each document is coded as the collection of some keywords extracted from the original document, and will directly be input to SOM, whereas each output layer node of SOM are coded as numerical vector as that of most Kohonen Networks. In order to directly separate documents into different groups, ring topology (Liu, wu, Liu, & wang, 2009) is adopted as our SOM structure, thus the number of groups can be any integral values. Like Kohonen Networks, it consists of two layers, input layer and competitive layer; the input layer receives input word vector representing a text, and a competitive layer indicating clusters; each node in this layer corresponds to one cluster (typically in SOM, a cluster may be formed from many nodes). Nodes in competitive layer are competitive with each other based on similarity between their weight vectors and input vectors; the winner node in that layer is the node with highest similarity between its corresponding weight vector and the given input vector.

The main contributions of this paper are as follows:

(1) Text Clustering has been extensively researched with a large number of machine learning and NLP-based techniques, most of them explored representing both document and neuron as vectors in one same high-dimension space, thus much computation time will be inevitably consumed. In many traditional SOM models, both documents and neurons are coded into high-dimension vector in one feature space. In this paper only neurons are coded into high-dimension vectors, which are same as many related works, whereas all documents are coded into sets of some words extracted from the original document. Thus both time complexity and space complexity can be reduced greatly. The difference between our approach with other methods such as PCA (Aguado, Montoya, Borras, Seco, & Ferrer, 2008; Sebzalli & Wang, 2001), LSI (Kontostathis & Pottenger, 2006; Wei, Yang, & Lin, 2008),random projection (Avogadri & Valentini, 2009; Lonardi, Szpankowski, & Yang, 2006) and co-occurrence (Morita et al., 2004; Tjhi & Chen, 2008) is that the features are not reduced or reformed semantically, thus need no extra preprocess time. Besides, these methods can also be combined with ours to achieve better efficiency.

(2) In order to solve the problem of computing similarity between document and neuron in our coding scheme, we give the corresponding similarity computation technologies, and the solution for mapping-error problem has also been proposed. The devised text clustering technique outperforms VSM + SOM across all benchmark data sets, as extensively verified through experiments. Especially when the size of documents is large, the actual efficiency improvements are more apparent. Thus it is suitable for applications where the documents to be processed are large and less clustering time is preferred, such as observing document sets from many different angels in short time or clustering results returned from searching engine.

(3) In addition, we also addressed the problem of incremental document clustering which allows for full utilization of previous clustering results and incremental reconstruction of SOM output layer with an emphasis on efficiency.

The main framework of our text clustering system is shown in Fig. 1.

The rest of this paper is organized as follows: Section 2 discuss the related works. Section 3 gives our adapted SOM model for text clustering; Section 4 introduces the feature coding method of both documents and neurons. Section 5 presents the fast similarity computation methods proposed; Section 6 discusses the incremental approach to clustering documents. then, in Section 7, the experiments and evaluation results are explained and discussed. Finally, we conclude and discuss future work in the last section.

2. Related works

SOM has been extensively applied in many areas for data clustering and achieved remarkable results. It is first proposed by professor Kohonen in Helsinki University (Kohonen & Kaski, 2000). SOM is an unsupervised-learning neural-network clustering method that produces a similarity graph of input data, which is much similar to the self-organizing characteristics of human brain. It consists of a finite set of models that approximate the open set of input data. The original SOM algorithm is a recursive regression process. A considerable amount of theoretical justifications is present in the literature to support SOM. It has been proven that, in text clustering research, SOM is one of the best learning algorithms (Huang, Ke, & Yang, 2008; Isa, Kallimani, & Lee, 2009).

The time complexity of SOM is O (k′m n), where k′is the number of neurons, m is the training times and n is the document number (m*n samples need to be inputted to train the network). As for the clustering of massive documents, Kaski and his colleagues developed famous WEBSOM system with SOM (Kaski, Honkela, Lagus, & Kohonen, 1998) in 1998. WEBSOM displays the results of text clustering graphically for browsing collection of documents. After that Kohonen and his colleagues have also implemented a modified version of WEBSOM to apply it more practically to a massive document collection in 2000 (Kohonen et al., 2000).

The efficiency of text clustering system depends closely on feature coding and similarity computation. Many SOM variants use VSM (Chow, Zhang, & Rahman, 2009; Pullwitt, 2002) to represent both documents and neurons, and calculate the similarity between them. Feature reduction or selecting features from text documents becomes one of the key issues in text clustering and received a fair amount of research with various methods. A systematic comparison study of dimension reduction techniques has been done by Tang, Bin Tang, Shepherd, Heywood, and Luo (2005) for text clustering problem. Sinka and David have proven that each document can be represented by only a few words (Sinka & Corne, 2005). They also argued that frequency is enough for represent documents and complex coding scheme may cost much more time. Currently in many text clustering systems, all documents and neurons are represented in high-dimension space (Hung, Chi, & Chen, 2009; Lee & Yang, 2009a; Lee & Yang, 2009; Martín-Guerrero & Palomares, 2006), thus the frequent similarity computation is very important for clustering efficiency.

3. Adapted SOM model with ring topology

In many SOM variants, e.g. in the WEBSOM, clusters were spread over several neurons and the partition of documents all depends on the density. The number of neurons in the map should be much larger than the perceived number of clusters in SOM. Whereas in our paper, we wish each neuron give a direct partition of input documents, I.e., each neuron represents one document class. The rectangular topology used by many SOM variants is hard to achieve this goal. For example, if there are 5 classes in the input documents, there usually need 2*3 = 6 neurons for rectangular topology, thus there will be one redundant neuron. Fig. 2. shows the ring output layer topology of SOM used in this paper. The advantage of this topology is that sector number (node number) can be any integers, and it will be possible to reflect topic distribution of the input documents more finely and make full use of neurons. Besides, the number of neighboring neurons for each neuron is same, thus it can guarantee the balance of networks and avoid edge effect which usually happens by using rectangular or hexagonal topology. In fact when there is need to expand output layer, neurons can be inserted gradually to avoid lack-of-use phenomenon of neurons.

Ring topology can be suitable for the partition of any size. If there is need to create new cluster in incremental clustering, the only thing to do is create one new neuron. Whereas for rectangular topology, at least one row or column will be inserted. In this paper, our focus in on fast clustering technology, thus the Number of clusters is supposed to be known beforehand and the network size has been set fixed. In this way we can mainly discuss the core technology of fast clustering. Besides, when it is need for dynamic clustering, neurons of any size can be inserted to reflect the input documents (Liu, Wang, & Liu, 2009). From the related research, there are many dynamic clustering methods (Fritzke, 1995; Rauber, Merkl, & Dittenbach, 2002; Zhou & Fu, 2005).The basic idea is to insert more neurons and make the network simulate input space.

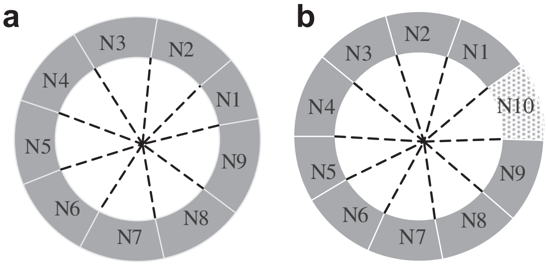

Fig. 2. The ring topology of V-SOM. (N10 in Fig. 2. (b) is the newly inserted neuron).

4. One novel document coding scheme for fast text clustering

Due to high-dimension features and many iterations of most text clustering algorithms, time cost of text clustering is remarkable. There are usually two skills to improve efficiency: efficient clustering algorithm and suitable coding method.

To achieve better clustering results, data model must accurately capture salient features of input data. In this paper some high-frequency keywords of each document are extracted to represent the original document, as we believe that these words can capture main contents of document, this can also be proven by many related works shown in Section 2. It is very critical to note that selecting a few words to represent each document will make number of the dimensions greatly reduce and save a lot of time for online stages. This is also partially where the efficiency of the model comes from. (See Fig. 3)

Most of the text clustering methods that are in use today are based on Vector Space Model (VSM) for representing document vectors (Aas et al., 1999; Hammouda & Kamel, 2004; Salton, Wong, & Yang, 1975; Salton & McGill, 1983; Salton, 1989).The advantage of VSM is that all the documents and neurons can be represented as vectors with same dimension number, thus the unstructured or semi-structured documents can be transferred into structured format, and provide possibility and convenience for calculating their similarity. Besides, Main machine learning algorithms such as NB (Naïve Bayes), SVM (Support Vector Machine), and MLP (Multilayer Perceptron) use numerical vectors as their input vectors. Each word in documents corresponds to a feature in the vector representation. This leads to high dimensionality of text documents. The main shortcoming of VSM applied in text clustering is that the computation will cost lots of time, especially when large-scale documents need to be processed. As similarity computation is very frequent, the high computation cost will become intolerable. Documents will usually be transferred into a bag of words, which can represent main contents of the original documents. For most text clustering algorithm using VSM, in order to represent documents in feature space, there are usually one stage to transfer bags of words into vectors in high-dimension space. This quantization step is computationally demanding.

For SOM text clustering, similarity computation between document and neuron is also frequent. In this paper we represent

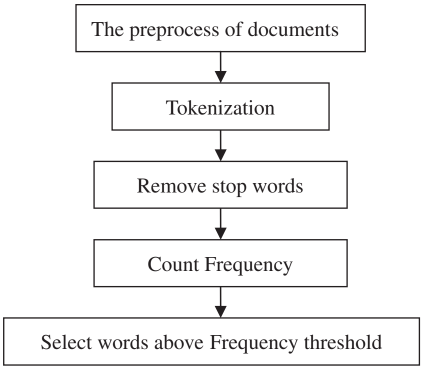
The preprocess of documents
Tokenization
Remove stop words
Count Frequency
Select words above Frequency threshold

Fig. 3. The process of text document.

document only as indexes of some keywords, other than the widely used high-dimension vectors which share same feature space with neurons. Thus both memory consummation and time to code documents can be reduced greatly. In such scenarios, two questions must be solved: one is how to construct feature space for neurons and how to represent neurons in this space; another is how to judge the similarity between neurons and documents as they are represented in different way.

Algorithm 1. Dynamic construction of feature space and representation of documents

```
Require: the keyword set of each document Keyword ($d_i$), the feature set vectorX (n) of documents $\mathbf{d}_0$,$\mathbf{d}_1$,. . . . . .,$\mathbf{d}_{n-1}$. Or vectorX (0) if no documents were processed previously.
$\mathbf{d_i}$← Next Document;
For each keyword $w_{ij}$ in Keyword ($\mathbf{d_i}$) do
    IF$w_{ij}$ is invectorX, THEN
        m ← the index of $w_{ij}$in vectorX;
        Add m to Vector ($\mathbf{d_i}$);
    ELSE
        add an element E to the end of vectorX, index{E, vectorX}++, word{E, vectorX} = $w_{ij}$;
        Add m to Vector ($\mathbf{d_i}$);
    END IF
End for
Output the elements of vectorX//feature sets for feature space;
Output the elements of Vector($\mathbf{d_i}$)//each document.
```

In this paper we use the following methods to construct feature space for neurons: as the documents to be processed are usually from open-domain, i.e., the topic of these documents can not be fixed, feature space will be generated dynamically according to these documents. The dimension of feature space is only from the documents because there is lack of domain lexicon or lexicons are too big compared with the number of dimensions generated. For example, suppose there are 3000 features generated from 1,000 documents, whereas the lexicon have 100,000 words, forming feature space by lexicon will cost great time. In this situation, selecting words from documents to form feature space will reduce the sparse elements in feature vectors and reduce redundancy of vector representation.

In such way, only neurons need to be represented as high-dimension vector, whereas the document will be coded as indexes of keywords. As number of document is usually much more than that of neuron and the basic operation of SOM is similarity computation between documents and neurons, thus computation cost will be degraded greatly. Another merit is that construction efficiency of feature space can also be improved. We use Algorithm 1 to construct feature space and code both documents and neurons.

The advantage of Algorithm 1 is that through one scan of all documents, both feature space construction and document coding can be done. In addition, it also can provide a good foundation for incremental clustering: when new batch of documents are added, the only thing to do is run the algorithm once by utilizing original feature space. Because documents are coded as indexes, it is convenient to compute the similarity between documents and neurons.

Our scenario in fact can also be helpful for many related research besides SOM clustering algorithm, such as *k*-means, AHC (Agglomerative Hierarchical clustering), and etc. Although there are some work which use string vector to represent the document feature such as NTSO (Taeho & Nathalie, 2005), the fast similarity computation and the rectification of wrong mapping problem are not solved in their work. The shortcoming of NTSO is that it need to construct similarity matrix beforehand, thus much extra time will be needed.

5. Fast similarity computation for text clustering

It is very frequent for SOM text clustering to compute similarity between neuron and document. Same as the work of standard SOM, each neuron vector is initialized as small elements. But in our scheme, neurons and documents are coded in different space, for one document $\mathbf{d_i}$, we use Algorithm 2 to find the most similar neuron.

Algorithm 2. Find the most similar neuron with document $\mathbf{d_i}$

```
Get the string vector ($\mathbf{d_i}$) of $\mathbf{d_i}$;
For each neuron $\mathbf{n_j}$,do;
    Get the corresponding Vector ($\mathbf{n_j}$) of $\mathbf{n_j}$;
    For each index element from vector ($\mathbf{d_i}$), denoted it as $k$, do;
        According to $k$, find the $k$th dimension for the corresponding numerical element $\mathbf{n_j}[k]$ of the Vector ($\mathbf{n_j}$) of neuron $\mathbf{n_j}$;
        Sum (j) = Sum (j)+$\mathbf{n_j}[k]$;
    End For;
End For;
For all value of $j$,do
    Find the maximum value of sum (j);
End For;
Select the corresponding neuron $\mathbf{N_{max}}$ as the wining neuron.
Output the similarity between document and neuron;
Update the weight of winning neuron and its neighboring neurons.
```

In Algorithm 2, we assume that all words extracted from each document are equally important. Thus for each document and neuron, the only thing to do is add together the weight of some elements which is located according to the indexes from string vector of document. For example, suppose the indexes of one document is {$\mathbf{f}_1$,, $\mathbf{f}_2$, $\mathbf{f}_3$,$\mathbf{f}_4$, $\mathbf{f}_5$}, then its similarity with neuron $\mathbf{n_j}$ can be calculated as Vector ($\mathbf{n_j}$)[$\mathbf{f}_1$]+Vector ($\mathbf{n_j}$)[$\mathbf{f}_2$]+Vector ($\mathbf{n_j}$)[$\mathbf{f}_3$]+Vector ($\mathbf{n_j}$)[$\mathbf{f}_4$]+Vector ($\mathbf{n_j}$)[$\mathbf{f}_5$]. Here $\mathbf{f_x}$ means the $\mathbf{f_x}$-th dimension of $\mathbf{n_j}$. Computing similarity in this way will cost much less time than cosine. According to the principle of SOM, we only need to find the most similar neuron with one input document, how much they are similar is not so important.

After the winner neuron $\mathbf{N_{winner}}$ has been found, which has the biggest similarity with the input document $\mathbf{d_i}$, $\mathbf{d_i}$ will be mapped onto$\mathbf{N_{winner}}$,$\mathbf{N_{winner}}$and its neighboring neurons will have the rights to adjust their weights of their dimensions which is indexed by elements of document vector of $\mathbf{d_i}$, i.e., add φ to the corresponding dimensions. For the above examples: Vector ($\mathbf{n_j}$)[$\mathbf{f}_1$] = Vector (**nj**)[$\mathbf{f}_1$]+ φ; Vector (**nj**)[$\mathbf{f}_2$] = Vector ($\mathbf{n_j}$)[$\mathbf{f}_2$]+φ; Vector ($\mathbf{n_j}$)[$\mathbf{f}_3$] = Vector (**nj**)[$\mathbf{f}_3$]+φ; Vector ($\mathbf{n_j}$)[$\mathbf{f}_4$] = Vector ($\mathbf{n_j}$)[$\mathbf{f}_4$]+φ; Vector ($\mathbf{n_j}$)[$\mathbf{f}_5$] = Vector ($\mathbf{n_j}$)[$\mathbf{f}_5$]+φ. Here the value of φ can be set through experimentation. After this, the documents which have the same topic with document $\mathbf{d_i}$will also have the biggest similarity with neuron$\mathbf{n_j}$, because the elements of $\mathbf{n_j}$ on these dimensions will be strengthened.

One problem to be solved is that, after one document **d** is input, if another document **d**′ is input again, and **d**′ has different topic from document **d**, then **d** and **d**′will be mapped onto one same neuron $\mathbf{n_j}$ by mistake because only a few dimensions will be adjusted in this method, and other dimensions of $\mathbf{n_j}$ may have not adjusted and thus document **d**′ may have biggest similarity with

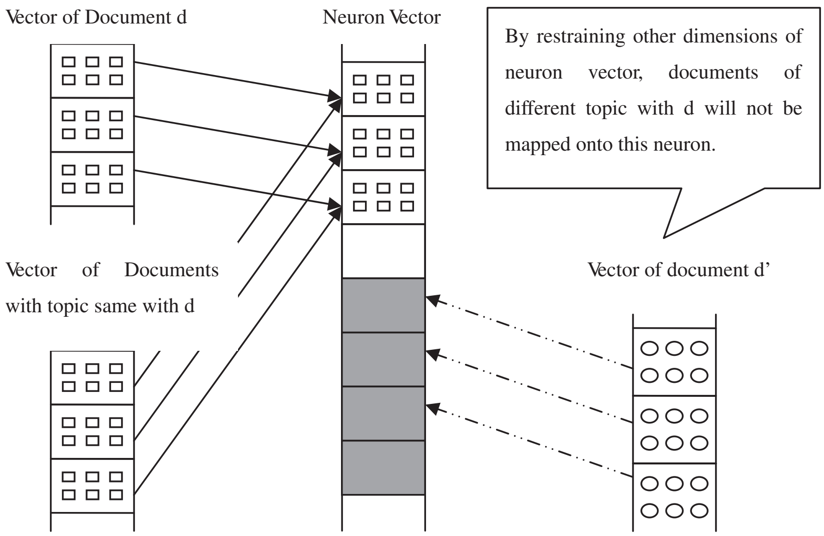
Vector of Document d
Neuron Vector
By restraining other dimensions of neuron vector, documents of different topic with d will not be mapped onto this neuron.
Vector of Documents with topic same with d
Vector of document d'

Fig. 4. Prevention of mapping error by restraining other dimensions.

neuron $\mathbf{n_j}$, as the other dimensions of neuron $\mathbf{n_j}$ are not adjusted. $\mathbf{n_j}$ still have the same chance to be selected as the winner. This situation will happen especially when the training just begins as most of dimensions of neurons have not been adjusted.

In order to overcome this problem, When add φ to some dimensions of neuron $\mathbf{n_j}$, we also subtract a value φ' to other dimensions. In this way, when document $\mathbf{d}'$ comes, which is about different topic, as other dimensions have smaller weight than those of other neurons, $\mathbf{n_j}$can not be selected as the most similar neuron with $\mathbf{d}'$. Whereas if another document $\mathbf{d}''$ comes, which share same topic with $\mathbf{d}$, as most of its dimensions may be same as document $\mathbf{d}$ and neuron $\mathbf{n_j}$, $\mathbf{d}''$ can be mapped onto neuron $\mathbf{n_j}$. This can be shown in Fig. 4. In addition, after one neuron has been adjusted, and has been again calculated as the most similar with one document, then one judgment will be done: if the dimensions to be modified have no or little intersections with the last adjustment, then the modification will be rolled back.

General speaking, document vector formed by standard VSM is high-dimension sparse matrix, and similarity computation in essence is the computation between high-dimension vectors. In our scheme features have not been subtracted or reduced, whereas the high computation cost has been avoided, this is where the high efficiency comes from. At the same time, we also give the corresponding solutions about the impact of this method may have on clustering quality.

6. Incremental clustering

For many online applications, input documents are dynamic, or time is a critical factor for usability, thus incremental clustering is an essential strategy (Agrawal, Han, Wang, & Yu, 2003; Agrawal, Han, Wang, & Yu, 2004; Kifer, Ben-David, & Gehrke, 2004; Zamir & Etzioni, 1999). This presents several challenges to traditional static clustering algorithms. Application examples include topic detection from a news stream, intrusion detection from continuous network traffic, and etc. (Sahoo & Callan, 2006; Sascha & Michael, 2006; O'Callaghan, Mishra, Meyerson, Guha, & Motwani, 2002).

This paper combines an efficient Incremental clustering algorithms with our feature coding and similarity computation strategy to achieve fast clustering of text documents. Incremental text clustering algorithms work by processing text documents one at a time, incrementally assigning documents to their respective clusters while they progress (Shi, 2005; Zhonghui, Junpeng, & Junyi, 2007).A "good" incremental clustering algorithm has to find the respective cluster for each newly introduced document without significantly sacrificing the accuracy of clustering due to insertion order or fixed object-to-cluster assignment.

Some incremental clustering methods include: GCS (Lihua, Lu, Jing, & Zongyong, 2005), ART (Hsu & Huang, 2008; Keskin, İlhan, & Özkan, 2010), Single-Pass Clustering (Asharaf & Narasimha Murty, 2003; Kärkkäinen & Fränti, 2007), K-Nearest Neighbor Clustering (González-Barrios & Quiroz, 2003), Suffix Tree Clustering (STC)(Han et al., 2006). Among them, Single-Pass Clustering algorithm basically processes documents sequentially, and compares each document to all existing clusters. If the similarity between the document and any cluster is above a certain threshold, then the document is added to the closest cluster; otherwise, it forms its own cluster. Usually, the method for determining the similarity between a document and a cluster is done by computing the average similarity of the document to all documents in that cluster.

Suppose there are $\mathbf{n}$ documents in all, $\mathbf{n}_1$ documents are selected as the base set to be clustered firstly, the remaining $\mathbf{n}_2(\mathbf{n}_2 = \mathbf{n} - \mathbf{n}_1)$ documents are selected as the objects for incremental clustering. There are usually three ways to process these $\mathbf{n}_1 + \mathbf{n}_2$ documents:

(1). Run the clustering system for these $\mathbf{n}_1 + \mathbf{n}_2$ documents in one time, which we call whole clustering, but the efficiency may be very low, as previous clustering results can not be fully utilized.

(2). Run the clustering system for $\mathbf{n}_1$ documents and $\mathbf{n}_2$ documents separately, and then merge their clustering results. Suppose there are $\mathbf{c}(\mathbf{n}_1)$ clusters for $\mathbf{n}_1$documents and $\mathbf{c}(\mathbf{n}_2)$ clusters for the newly added $\mathbf{n}_2$ documents, then the whole text clustering system will be run twice, and additional computation will also be done to merge these $\mathbf{c}(\mathbf{n}_1)$ clusters and $\mathbf{c}(\mathbf{n}_2)$ clusters. As in our coding scheme, both clusters are high-dimension vectors, thus the computation will also be high.

(3). Utilize the clustering model formed by the $\mathbf{n}_1$ documents (similar to Single-Pass Clustering), classify the remaining $\mathbf{n}_2$ documents, and add them to some clusters, whereas for the document which cannot be classified, create one new neuron and assign the document to it. The time complexity is that of $\mathbf{n}_1$ documents plus the complexity of the similarity computation between $\mathbf{n}_2$ and the clusters. The complexity of latter will be relatively low due to our similarity computation method demonstrated in Section 4.

In this paper, we adopt the third as our incremental clustering algorithm, as it is more efficient in our feature coding scheme. The details are shown in Algorithm 3.

Algorithm 3. incremental clustering

```
1. $\mathbf{L_n}$← {neuron list created firstly}; $\mathbf{L_d}$← {list of incremental documents};
2. **For** each document **d** in $\mathbf{L_d}$, **do**
3. **For** each neuron **n** in $\mathbf{L_n}$, **do**
4. Compute the similarity **similarity**(**d**,**n**) between $d$ and $n$;
5. Find the neuron $\mathbf{n_{max}}$ which has the biggest similarity with $d$;
6. **If similarity**(**d**,$\mathbf{n}_{\max}$)> threshold $\mathbf{s_t}$,**then**
7. Add **d** to $\mathbf{n_{max}}$, delete **d** from $\mathbf{L_d}$;
8. **Else then**//d was not added to any cluster
9. Create a new neuron $\mathbf{n_{new}}$;
10. Add $\mathbf{n_{new}}$ to $\mathbf{L_n}$;
11. Add **d** to $\mathbf{n_{new}}$, delete **d** from $\mathbf{L_d}$;
12. **End if**
13. **End for**
14. **End for**
```

The incremental clustering algorithm in Algorithm 3 are based on the above framework (see Sections 3 and 4). The algorithm works incrementally by receiving a new document, and for each cluster calculates their similarity. the cluster with the biggest similarity will be selected, if the similarity value is greater than threshold, the document will be assigned into that cluster, otherwise one new neuron will be created, and assign that document to it.

In this paper, as there are lots of documents to be processed, we combine both sampling technology, incremental clustering technology and our feature coding scheme to improve the clustering efficiency further. Some documents (at least 50%) are selected to construct clustering model, and the remaining documents will be classified or clustered according to the clustering model constructed. It is believed that by this way the clustering quality will not be affected so much, whereas the efficiency will be greatly improved.

7. Experimental results and discussion

7.1. Experimental setup

The experimental evaluation was performed on two different data sets: DS1 and DS2 separately. The first data set: DS1 is a subset of the full 20-newsgroups collection of USENET news group articles, which is the widely used text categorization datasets. Each news group constitutes a different category, with varying overlap between them. DS2 is news documents crawled from web, there are much documents in DS2 so that the number of Categories can not be easily determined manually. Since they are unlabeled, this collection will be used to examine our system efficiency for large-scale text clustering. Some clusters generated will be selected randomly to be reviewed by human to evaluate its performance. (See Table 1)

Table 1
Data Sets Description.

Dataset	Descriptions	# Of Docs.	Categories	# of Dims
DS1	20 Newsgroups	1000	20	1798
DS2	Documents crawled from web	100,000	N/A	10892

7.2. Evaluation methodology

In our approach, we use hard clustering where we assign only one cluster to each input document. The first and important evaluation measure in this paper is the CPU time consumed and another is clustering quality. In this paper the run time results were measured on a PC with Intel Pentium IV 1.6 GHz CPU and 1 GB memory, running Windows XP.

The first clustering quality measurement is F-measure (Massey, 2005). The higher the overall F-measure, the better the clustering, due to the higher accuracy of clusters generated mapping to the original classes. For one generated cluster $r$and one predefined class s:

$$recall(r,s) = n(r,s)/n_s \tag{1}$$

$$precision(r,s) = n(r,s)/n_r \tag{2}$$

$n(r,s)$ is document number of the intersection between r and s. n_r is document number of cluster r, n_s is document number of class s. F measure between cluster r and class s can be calculated as

$$F(r,s) = (2^*recall(r,s)^*precision(r,s))/((precison(r,s) + recall(r,s)) \tag{3}$$

The overall F measure can be calculated as

$$F = \sum_i \frac{n_i}{n} \max\{F(i,j)\} \tag{4}$$

Here n is the number of all documents. $n_i$is document number of class i.In this paper, we use F measure for the clustering results of DS1.

Because the size of DS2 is big, we select accuracy as the evaluation method. Suppose there are m clusters are generated (for DS2, several thousands of clusters may be generated), we select m'(in this paper, we set m' = 100) sample clusters from them to judge the overall clustering quality. Suppose there are # $of(c_i)$ documents in cluster c_i, which is one of these m' clusters, and the class $C_k$which has the biggest document number in c_i, denoted as $\#of(c_i,C_k)$ will be assumed as the dominant topic of this cluster, thus the accuracy of cluster $c_i$can be calculated as

$$Accuarcy(c_i) = \#of(c_i, C_k)/\#of(c_i). \tag{5}$$

The average accuracy of m'clusters,$Accuracy(m')$, which will be used to judge the overall quality, can be calculated in the following way:

$$Accuracy(m') = \frac{1}{m'} \sum_{i=0}^{m'-1} Accuracy(c_i) \tag{6}$$

We tested the effectiveness of our text representation model in 7.3, the comparison of clustering results and the effect of incremental document clustering in 7.4. The analysis of clustering results and the help for cognition are conducted in 7.5.

7.3. The experimental comparison of feature representation methods

Feature representation and feature quantization is a necessary step for many related works. Their basic logics are: first construct feature space using the keywords extracted from each document, and then represent each document as one vector in this space. Whereas in this paper, documents are directly coded as indexes of keywords. The time consumed in both these methods are compared, the results are shown in Fig. 5 (the coding method for neurons of SOM is same as that of related work). The contrast of time consumed for similarity computation is shown in Table 2 (The second column is the traditional high-dimension cosine computation in VSM mode). It can be seen that both the efficiency of document coding and similarity computation are improved greatly. For similarity computation, our approach cost much less time compared with traditional way, thus will be very helpful for fast clustering as similarity computation is a very frequency operation.

7.4. Comparison of clustering results

We first tested the F measure performance of our system on DS1. The results are shown in Figs. 6 and 7 respectively. We compare our system with another text clustering system GHSOM (http, xxxx), which is an open-source widely-known program and also used for text clustering. It can be seen that the efficiency of our approach is higher than GHSOM due to our novel coding scheme and similarity computation. Besides, as we adopted ring-topology for SOM, which will give a direct partition for documents, we also achieved relatively higher F measure than GHSOM which is rectangular topology.

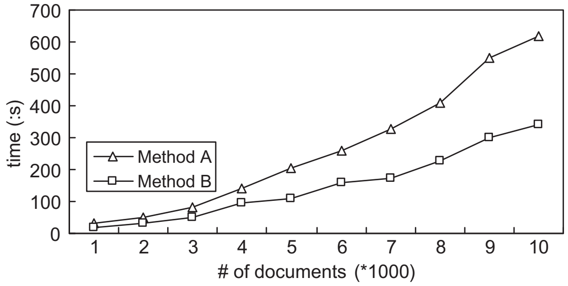

Fig. 5. The comparison of CPU time consumed for document coding (dataset: DS2. Method A: VSM; Method B: our approach).

Table 2
The contrast of time consumed for similarity computation (dataset: DS2. # of documents = 10000, # of features = 4329).

# of computation	VSM (:s)	Our approach (:s)
50, 000	4.70	0.13
100, 000	9.42	0.19
200, 000	18.82	0.21
300, 000	28.18	0.26
500, 000	47.26	0.32
800, 000	75.09	0.47
1000, 000	93.98	0.63
2000, 000	187.64	2.25

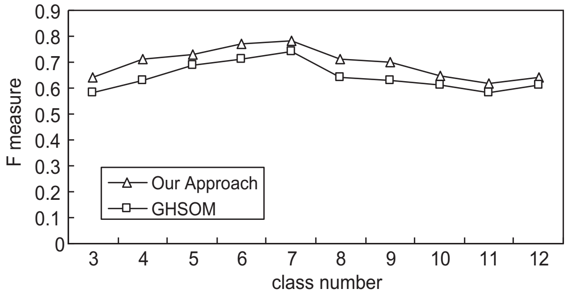

Fig. 6. Clustering quality on DS1 (F measure).

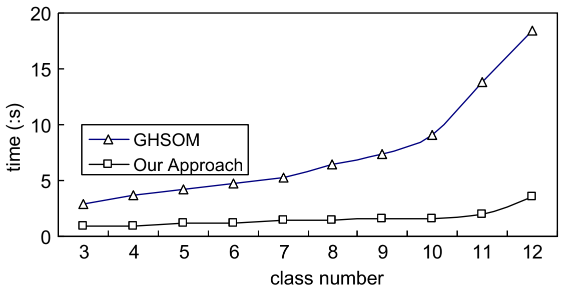

Fig. 7. Clustering time on DS1 (: s).

To observe the performance of our approach on bigger datasets, we tested it on dataset DS2, and use accuracy as the evaluation measurement. This time, we also manipulate DS2 data set to get two different parts of documents with different proportion. The first one, part 1, is selected randomly as the whole clustering, i.e., they will be first clustered and some clusters can be formed firstly. The second part, part 2, is the remaining documents, will be the objects of the incremental clustering. The values of Proportion are varied so that we can study the effect. The results are shown in Tables 3 and 4 separately. Table 3 is the result of 10,000 documents which is part of DS2, and Table 4 is all documents of DS2.

Several observations can be made from the results.1) First of all, the clustering quality with proportion equal 50% is the worst performing one in all situations When the proportion is 50%, as the documents for whole clustering is very few, thus it is hard to assign the remaining documents, and many of them will create one new neuron. The time consumed is also much. As 50% documents can not cover most words, thus it is difficult to classify the remaining documents, and the clustering quality will be affected. In contrast, a proportion value of $0.8 \sim 0.85$ leads to much better balance of quality and efficiency; (2). When documents are large-scale or the clustering computation is frequent, the method with high efficiency become more urgent, and there are a bigger time gap

Table 3
Text clustering performance in terms of accuracy (Dataset: 10000 documents from DS2; Method A: VSM; Method B: Our approach).

Ratio of part1/ part2	# of features	% of lost features	Time of method A (: s)	Accuracy of method A	Time of method B (: s)	Accuracy of method B
100%, 0%	4329	0.00%	1492.58	0.88	162.87	0.87
95%, 5%	4285	1.02%	1386.53	0.87	159.28	0.86
90%, 10%	4249	1.85%	1243.97	0.87	154.53	0.85
85%, 15%	4218	2.56%	1189.32	0.85	149.67	0.85
80%, 20%	4187	3.28%	1061.42	0.83	137.15	0.82
75%, 25%	4146	4.23%	1002.38	0.79	121.20	0.78
70%, 30%	4119	4.92%	891.63	0.75	118.76	0.73
65%, 35%	4068	6.03%	823.78	0.74	107.70	0.70
60%, 40%	4031	6.88%	763.67	0.69	96.75	0.65
55%, 45%	3982	8.02%	712.19	0.67	87.71	0.62
50%, 50%	3929	9.24%	651.52	0.67	79.96	0.61

Table 4
Text clustering performance in terms of accuracy (dataset: all documents of DS2; Method A: VSM; Method B: Our approach).

Ratio of part1/ part2	# of features	% of lost features	Time of method A (: s)	Accuracy of method A	Time of method B (: s)	Accuracy of method B
100%, 0%	10892	0.00%	37953.43	0.87	3584.62	0.86
95%, 5%	9780	0.83%	35721.89	0.85	3385.21	0.85
90%, 10%	9559	2.10%	33946.42	0.84	3106.66	0.85
85%, 15%	9322	3.55%	31647.81	0.83	2963.64	0.80
80%, 20%	9016	4.67%	29352.59	0.81	2706.12	0.79
75%, 25%	8869	6.23%	28016.06	0.79	2493.25	0.77
70%, 30%	8687	8.15%	25924.52	0.77	2267.10	0.76
65%, 35%	8491	10.22%	23743.23	0.75	2014.54	0.72
60%, 40%	8293	12.32%	20694.84	0.72	1792.93	0.66
55%, 45%	8058	14.86%	18917.32	0.67	1581.50	0.62
50%, 50%	7756	17.99%	16902.71	0.58	1333.19	0.54

saved in our approach; (3) Overall, these results suggest a general intuitive fact in incremental text clustering. When efficiency is the preferred measurement, utilizing the formed clustering model will further reserve a lot of time.

7.5. The analysis of clustering results and the help for cognition

As shown before, the objective of text clustering is by organization of massive documents, make people quickly find the documents they need. We designed the following methods to evaluate cognition efficiency: (1). through retrieval (local). For all N documents, select N' documents and represent them to users. After that, let users find these N' documents from the clustering results, and record the time used. We assume that the less time used, the better the clustering quality is; (2).Through recall (global). For N documents, select N' documents and represent them to users, then give users $N' + N''$documents (N''documents are totally different from N documents), and let users find out theN' documents according to recall. Suppose the correct number is N'_c, then the accuracy is N'_c/N'. The results are shown in Figs. 8 and 9 separately.

The above two methods can evaluate two functions of text clustering system. Sometimes we want to find documents we need; sometimes we expect to know what these documents are about. It can be demonstrated that from the beginning, as when document number is small, the difference is not clear, as the effect of clustering can not emerge, and the semantic border of different documents can still be remembered by people, this conforms to short-time remember in cogitation science; when there are more documents, the difference becomes much more apparent, as by clustering, the semantic partition of documents make people can remember more content.

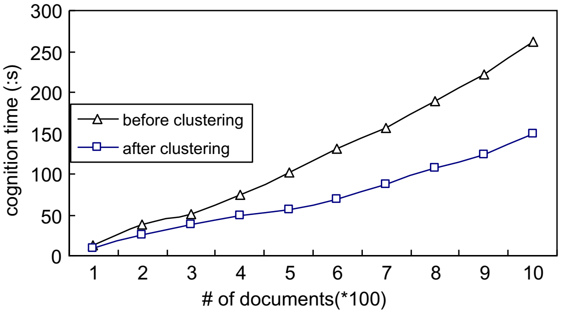

Fig. 8. The evaluation of cognition (retrieval, $N' = 5$).

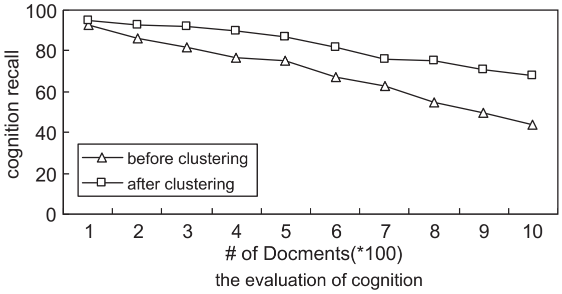

Fig. 9. The evaluation of cognition (recall, $N' = 10$; $N'' = 10$).

8. Conclusions and future research

In this paper, we investigated an efficient feature coding algorithm and similarity computation skills and constructed an efficient text clustering techniques by combining it with existing SOM clustering algorithms and coding schemes. The feature coding method which employs indexes of string sets to represent the document outperforms other approaches, and enables us to perform similarity calculation between documents in a very efficient and accurate way. The quality of clustering achieved significantly surpasses the traditional vector space model based approaches. We have shown that the proposed coding algorithm can capture the main features of the original document while reducing the redundant information greatly, thus clustering efficiency can be improved. The method was extensively tested with numerous experiments using well-known benchmark data sets and compared with exact experimental settings against traditional way.

There are a number of future research directions to extend and improve this work. One direction that this work might continue on is to improve on the accuracy of similarity calculation between documents by employing different similarity calculation strategies. Although our current scheme proved more efficient than VSM representation. there is still room for improvement. Although the work presented here is aimed at SOM clustering and some SOM advantage has been talked, it could be easily adapted to any other algorithm such as k-means as well. Our intention is to investigate the usage of such model on other algorithm or other high-dimension data clustering and see its effect on clustering effect compared to traditional methods.

Acknowledgments

This work was supported by Chinese 863 program (2007AA01Z172) and the National Natural Science Foundation of China(70773029 and 60603092).

References

Aas, K. & Eikvil, L. (1999). *Text categorisation: A survey*. Technical Report 941, Norwegian Computing Center.

Aggarwal, Yu, C. C., & Philip S. (2006). A framework for clustering massive text and categorical data streams. In *Proceedings of the sixth SIAM international conference on data mining* (pp. 479–483.

Agrawal, C. C., Han, J., Wang, J., & Yu, P. S. (2003). A framework for clustering evolving data streams. In *Proceedings of 29th international conference on very large data bases. Berlin, Germany* (pp. 81–92).

Agrawal, C. C., Han, J., Wang, J., & Yu, P. S. (2004). A framework for projected clustering of high dimensional data streams. In *Proceedings of 30th international conference on very large data bases. Toronto, Canada* (pp. 852–863).

Aguado, D., Montoya, T., Borras, L., Seco, A., & Ferrer, J. (2008). Using SOM and PCA for analysing and interpreting data from a P-removal SBR. *Engineering Applications of Artificial Intelligence, 21*(6), 919–930.

Aliguliyev, R. M. (2009). Clustering of document collection – A weighting approach. *Expert Systems with Applications, 36*(4), 7904–7916.

Aliguliyev, R. M. (2009). Clustering of document collection – A weighting approach. *Expert Systems with Applications, 36*(4), 7904–7916.

Asharaf, S., & Narasimha Murty, M. (2003). An adaptive rough fuzzy single pass algorithm for clustering large data sets. *Pattern Recognition, 36*(12), 3015–3018.

Avogadri, R., & Valentini, G. (2009). Fuzzy ensemble clustering based on random projections for DNA microarray data analysis. *Artificial Intelligence in Medicine, 45*(2-3), 173–183.

Chow, T. W. S., Zhang, H., & Rahman, M. K. M. (2009). A new document representation using term frequency and vectorized graph connectionists with application to document retrieval. *Expert Systems with Applications, 36*(10), 12023–12035.

Chow, T. W. S., Zhang, H., & Rahman, M. K. M. (2009). A new document representation using term frequency and vectorized graph connectionists with application to document retrieval. *Expert Systems with Applications, 36*(10), 12023–12035.

Cios, K., Pedrycs, W., & Swiniarski, R. (1998). *Data mining methods for knowledge discovery*. Boston: Kluwer Academic Publishers.

Fritzke, B. (1995). Growing grid—A self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, 9–13.

González-Barrios, J. M., & Quiroz, A. J. (2003). A clustering procedure based on the comparison between the k nearest neighbors graph and the minimal spanning tree. *Statistics & Probability Letters, 62*(1), 23–34.

Hammouda, K. M., & Kamel, M. S. (2004). Efficient phrase-based document indexing for web document clustering. *IEEE Transactions On Knowledge And Data Engineering, 16*(10), 1279–1296.

Han, S., Lee, S. G., Kim, K.-H., Choi, C. J., Kim, Y. H., & Hwang, K. S. (2006). CLAGen: A tool for clustering and annotating gene sequences using a suffix tree algorithm. *Biosystems, 84*(3), 175–182.

Hsu, C.-C., & Huang, Y.-P. (2008). Incremental clustering of mixed data based on distance hierarchy. *Expert Systems with Applications, 35*(3), 1177–1185.

<http://www.ifs.tuwien.ac.at/~andi/somlib/download/index.html>

Huang, S.-H., Ke, H.-R., & Yang, W.-P. (2008). Structure clustering for Chinese patent documents. *Expert Systems with Applications, 34*(4), 2290–2297.

Hung, C., Chi, Y.-L., & Chen, T.-Y. (2009). An attentive self-organizing neural model for text mining. *Expert Systems with Applications, 36*(3), 7064–7071. Part 2.

Hung, C., Chi, Y.-L., & Chen, T.-Y. (2009). An attentive self-organizing neural model for text mining. *Expert Systems with Applications, 36*(3), 7064–7071. Part 2.

Isa, Dino, Kallimani, V. P., & Lee, Lam Hong (2009). Using the self organizing map for clustering of text documents. *Expert Systems with Applications, 36*(5), 9584–9591.

Kärkkäinen, I., & Fränti, P. (2007). Gradual model generator for single-pass clustering. *Pattern Recognition, 40*(3), 784–795.

Kaski, S., Honkela, T., Lagus, K., & Kohonen, T. (1998). WEBSOM-self organizing maps of document collections. *Neurocomputing, 21*, 101–117.

Keskin, G. A., İlhan, S., & Özkan, C. (2010). The fuzzy ART algorithm: A categorization method for supplier evaluation and selection. *Expert Systems with Applications, 37*(2), 1235–1240.

Kifer, D., Ben-David, S., & Gehrke, J. (2004). Detecting changes in data streams. In *Proceedings of 30th international conference on very large data bases. Toronto, Canada* (pp. 180–191).

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics, 43*(1), 59–69.

Kohonen, T., Kaski, S., et al. (2000). Self organization of a massive document collection. *IEEE Transaction on Neural Networks, 11*(3), 574–585.

Kohonen, T., Kaski, S., Lagus, K., Salojarvi, J., Paatero, V., & Saarela, A. (2000). Self organization of linguistic features and clustering algorithms for topic document clustering. *The Proceedings of 231 ACM SIGIR*, 224–231.

Kontostathis, A., & Pottenger, W. M. (2006). A framework for understanding Latent Semantic Indexing (LSI) performance. *Information Processing & Management, 42*(1), 56–73.

Lee, C.-H., & Yang, H.-C. (2009). Construction of supervised and unsupervised learning systems for multilingual text categorization. *Expert Systems with Applications, 36*(2), 2400–2410. Part 1.

Lee, C.-H., & Yang, H.-C. (2009). Construction of supervised and unsupervised learning systems for multilingual text categorization. *Expert Systems with Applications, 36*(2), 2400–2410. Part 1.

Lee, C.-H., & Yang, H.-C. (2009a). Construction of supervised and unsupervised learning systems for multilingual text categorization. *Expert Systems with Applications, 36*(2), 2400–2410. Part 1.

Lihua, Wu, Lu, Liu, Jing, Li, & Zongyong, Li (2005). Modeling user multiple interests by an improved GCS approach. *Expert Systems with Applications, 29*(4), 757–767.

Liu, Y., Cai, J., Yin, J., & Huang, A. (2006). An efficient clustering algorithm for small text documents. In *Seventh international conference on web-age information management workshops*.

Liu, Y., Wang, X., & Liu, M. (2009). V-SOM: A text clustering method based on dynamic SOM model. *Journal of Computational Information Systems, 5*(1), 141–145.

Liu, Y., wu, C., Liu, M., & wang, X. (2009). V-SOM: A text clustering method based on dynamic SOM model. *Journal of computational information systems, 5*(1), 141–145.

Lois, R., Olivier, C., & Francois, Y. (2007). Inference and evaluation of the multinomial mixture model for text clustering. *Information Processing and Management, 43*(5), 1260–1280.

Lonardi, S., Szpankowski, W., & Yang, Q. (2006). Finding biclusters by random projections. *Theoretical Computer Science, 368*(3), 217–230.

Martín-Guerrero, J. D., & Palomares, A. (2006). Studying the feasibility of a recommender in a citizen web portal based on user modeling and clustering algorithms. *Expert Systems with Applications, 30*(2), 299–312.

Louis, M. (2005). Evaluating and comparing text clustering results. In *Proceedings of the IASTED international conference on computational intelligence* (pp. 85–90).

Morita, K., Atlam, E., Fuketra, M., Tsuda, K., Oono, M., & Aoe, J. (2004). Word classification and hierarchy using co-occurrence word information. *Information Processing & Management, 40*(6), 957–972.

O'Callaghan, L., Mishra, N., Meyerson, A., Guha, S., & Motwani, R. (2002). Streaming-data algorithms for high-quality clustering. In *Proceedings of 18th international conference on data engineering. San Jose, CA* (pp. 685–696).

Pullwitt, Daniel (2002). Integrating contextual information to enhance SOM-based text document clustering. *Neural Networks, 15*(8-9), 1099–1106.

Rauber, A., Merkl, D., & Dittenbach, M. (2002). The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks, 13*(6), 1331–1341.

Sahoo, N., Callan, J., et al. (2006). Incremental hierarchical clustering of text documents. In *International conference on information and knowledge management* (pp. 357–366).

Salton, G. (1989). Automatic text processing. In *The transformation, analysis,and retrieval of information by computer*. Reading, Mass: Addison Wesley.

Salton, G., & McGill, M. J. (1983). Introduction to modern information retrieval. In *McGraw-Hill computer science series*. New York: McGraw-Hill.

Salton, G., Wong, A., & Yang, C. (1975). A vector space model for automatic indexing. *Communications of the ACM, 18*(11), 613–620.

Salton, G., wong, A., & Yang, C. (1975). A vector space model for automatic indexing. *Communications of the ACM, 18*(11), 613–620.

Saraçoğlu, R., Tütüncü, K., & Allahverdi, N. (2007). A fuzzy clustering approach for finding similar documents using a novel similarity measure. *Expert Systems with Applications, 33*(3), 600–605.

Saraçoğlu, R., Tütüncü, K., & Allahverdi, N. (2008). A new approach on search for similar documents with multiple categories using fuzzy clustering. *Expert Systems with Applications, 34*(4), 2545–2554.

Sascha, H., & Michael, W. (2006). Incremental clustering of newsgroup articles. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*, 332–341.

Sebzalli, Y. M., & Wang, X. Z. (2001). Knowledge discovery from process operational data using PCA and fuzzy clustering. *Engineering Applications of Artificial Intelligence, 14*(5), 607–616.

Shi, Z. (2005). *Efficient Streaming Text Clustering Source: Neural Networks, 18*(5–6), 790–798.

Sinka, M. P., & Corne, D. W. (2005). The BankSearch web document dataset: Investigating unsupervised clustering and category similarity. *Journal of Network and Computer Applications, 28*(2), 129–146.

Song, W., Li, C. H., & Park, Soon Cheol (2009). Genetic algorithm for text clustering using ontology and evaluating the validity of various semantic similarity measures. *Expert Systems with Applications, 36*(5), 9095–9104.

Taeho, J. & Nathalie, J. (2005). Text clustering with NTSO. In *Proceedings of the international joint conference on neural networks* (pp. 558–563).

Tang, B., Shepherd, M., Heywood, M. I., & Luo, X. (2005). Comparing dimension reduction techniques for document clustering. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 292–296.

Tjhi, W.-C., & Chen, L. (2008). A heuristic-based fuzzy co-clustering algorithm for categorization of high-dimensional data. *Fuzzy Sets and Systems, 159*(4), 371–389.

Wei, C.-P., Yang, C. C., & Lin, C.-M. (2008). A latent semantic indexing-based approach to multilingual document clustering. *Decision Support Systems, 45*(3), 606–620.

Zamir, O., & Etzioni, O. (1999). Grouper: A dynamic clustering interface to Web search results. *Computer Networks, 31*(11-16), 1361–1374.

Zhonghui, F., Junpeng, B., & Junyi, S. (2007). Incremental algorithm of text soft clustering Source: Hsi-An Chiao Tung Ta Hsueh. *Journal of Xi'an Jiaotong University, 41*(4), 398–401. +411.

Zhou, J., & Fu, Y. (2005). Clustering high-dimensional data using growing SOM. *Advances in Neural Networks (Lecture Notes in Computer Science)*, 63–68.